Document Control No. ECPN SDD.2

Software Development Document

(SDD)

for the ECPN System

Version 1.0.6

10/17/96

Contract No. F19628-93-D-0019
CDRL Sequence No. B003

Prepared for:

Defense Information Systems Agency (DISA)
45335 Vintage Park Plaza
Sterling, VA 20166-6701

Prepared by:

Inter-National Research Institute, Inc.
12350 Jefferson Avenue, Suite 380
Newport News, Virginia 23602

Document Control No. ECPN SDD.2

Software Development Document

(SDD)

for the ECPN System

Version 1.0.6

10/17/96

Prepared for:

Defense Information Systems Agency (DISA)
45335 Vintage Park Plaza
Sterling, VA 20166-6701

Prepared by:

Inter-National Research Institute, Inc.
12350 Jefferson Avenue, Suite 380
Newport News, Virginia 23602

Authenticated by ______________________________ Approved by ______________________________
(Contracting Agency) (Contractor)

Date____ Date ____________________________________

Table of Contents

List of Appendices

Section 1
Scope

This Software Design Document (SDD) applies to applies to Electronic Commerce Processing Node (ECPN), which is a Computer Software Configuration Item (CSCI) of the system identified as Electronic Commerce/Electronic Data Interchange (EC/EDI). This document follows the standards set forth in *Military Standard Software Development and Documentation* (MIL-STD-498) and in the Data Item Description (DID) for a Software Design Document (DI-IPSC-81435), as tailored by INRI. See Appendix A for a description of the document's tailoring.

1.1 Identification

The system identifier for ECPN has not yet been assigned. This document applies to Version 1.0.6 of the software. The purpose of ECPN is to provide NEP/Gateway enhancement to a platform environment that has more capability and functionality. The enhancement must ensure interoperability, economies of scale, and compliance to standards by the Department of Defense (DoD) and Federal Program Management Office (PMO). The current NEP and Gateway architecture and functionality are to be merged and redesignated as an ECPN.

1.2 System Overview

ECPN is being developed by Inter-National Research Institute (INRI) specifically for the EC/EDI system. The role of ECPN is to enhance the current EC/EDI NEP/Gateway system. The fundamental objectives of this effort are to:

- Maintain rigorous accountability end-to-end within the NEP/Gateway processing, with no single point of failure that could cause loss or non-delivery of data.
- Automate the processes required to place DISA EC/EDI into a high volume production environment; should include periodic automated reconciliation mechanisms to ensure that no deliveries are missed.
- Eliminate the UNIX® scripts and provide enhanced functionality in executable code.
- Enhance NEP/Gateway functionality by providing a transition from batch store/forward capability to single transaction mode.
- Implement basic ORACLE® RDBMS archival capability.
- Provide for backup archival, information retrieval, usage reporting and audit trails.
- Provide basic retransmission and recovery as well as status monitoring.
- Provide for automated notification of communication failure and restore and provide status monitoring.

1.3 Document Overview

This document describes the design for the ECPN CSCI of the EC/EDI system, including the allocation of requirements to the Computer Software Units (CSUs) that comprise ECPN.

The following is an overview of each section of this SDD:

Section 1	Scope	States the purpose of the EC/EDI system; describes the role of ECPN within EC/EDI; and states the purpose of this SDD.
Section 2	Referenced Documents	Lists the documents applicable to this SDD.
Section 3	CSCI-wide Design Decisions	Addresses ECPN's behavioral design and the selection and design of the software units that make up this CSCI.
Section 4	CSCI Architectural Design	Identifies the software units that comprise ECPN and the concept of execution among these units.
Section 5	CSCI Detailed Design	Describes the design decisions and any constraints associated with each software unit of ECPN.
Section 6	Requirements Traceability	Describes the traceability between and among ECPN requirements in this SDD and the ECPN system requirements.
Section 7	Notes	Defines the acronyms and abbreviations used in this SDD.

Section 2
Referenced Documents

The following documents are referenced in this SDD. In the event of a later version of a referenced document being issued, the later version shall supersede the referenced version.

- *Data Item Description - Software Design Document* (DI-IPSC-81435), April 1989.
- *Military Standard Software Development and Documentation* (MIL-STD-498), Department of Defense, December 1994.
- Software Requirements Specification (SRS) for the ECPN System, Version 1.0.6, February 1996.

Section 3
CSCI-Wide Design Decisions

Decisions about ECPN's behavioral design and the selection and design of the software units that make up this CSCI are addressed in Section 3.2 of the Software Requirements Specification (SRS) for the ECPN System.

Section 4
CSCI Architectural Design

This section describes the following architectural design elements of the ECPN CSCI:

- CSCI components
- Concept of execution
- Interface design

4.1 ECPN Software Units

This section:

- ◆ Identifies the software units that make up the ECPN.
- ◆ States the purpose of each software unit.
- ◆ Identifies each software unit's development status/type (such as new development, existing design or software to be reused as is, existing design or software to be reengineered, software to be developed for reuse, software planned for Build N, etc.). For existing design or software, the description provides identifying information, such as name, version, documentation references, library, etc.
- ◆ Identifies the program library in which the software that implements each software unit is to be placed.

4.1.1 Passive Communications

Purpose - Passive communications are accomplished when a message is received at the ECPN without any explicit action by the ECPN. Passive communications are receive only. Passive communication is handled by assigning local UNIX user accounts to receive messages from remote sites. There are two types of passive communications - electronic mail and file transfer protocol (ftp). The mechanism by which the messages populate the local file system is dependent on the communication type.

Development Type - New development

Library - Implemented in the EmailRecv and Unix ftp applications

4.1.2 Active Communications

Purpose - The Active Communications Unit is used to accomplish two-way communications between the ECPN and other participating nodes. It is composed of communication interfaces that are configurable through the Interface Database Unit. The execution of the interfaces is controlled and monitored by a communications server. The interfaces perform message input and output operations on the physical device assigned to the interface (i.e., serial line, network).

Development Type - Extension of existing software from Unified Build Version 2.3.0.1

Library - Implemented in the FtpComms, EmailSend, and NEPComms applications and the Process Control Manager server

4.1.3 Check Login

Purpose - The Check Login Unit is a utility used by the system to check for the presence of remote user logins and to act upon data inserted into user accounts by remote users. The user directories to be acted upon are determined by information contained in the Remote User Database Unit, designating user login names, user directions, user type, and user activity.

Development Type - New development

Library - Implemented in the CheckLogin application

4.1.4 Archiver

Purpose - The Archiver archives a given buffer or file to a data file on disk without altering or translating the original buffer or file. This provides an effective way to accurately archive a message's raw data, character by character, resulting in an exact copy of the data transmitted or received.

Development Type - New development

Library - Implemented in the libBatchArch library

4.1.5 X12 Translator

Purpose - The X12 Translator Unit is intended to translate raw X12 messages into a standard representation that can be logged, decoded and routed by ECPN. There are two functions performed in the X12 Translation: segment terminator translation and X12 message delimiting. Segment terminator translation references the Segment Terminator Database Unit to standardize the byte representation of the raw message. X12 message delimiting separates a batch of one or more incoming messages into individual messages so they can be processed by ECPN.

Development Type - Extension of existing software from Unified Build Version 2.3.0.1

Library - Implemented in the libComms and libX12Conv library archives and the GenImportFile server

4.1.6 UDF Translator

Purpose - The UDF Translator Unit is intended to translate User Defined Format (UDF) messages to ANSI X12 standard messages and X12 standard messages to UDF messages. The type of translation is defined by the type of message and origin (for incoming traffic) and type of message and destination (for outgoing traffic).

Development Type - New development

Library - Implemented in the libXlation library archive and the GenImportFile server

4.1.7 Incoming Communications Manager

Purpose - The Incoming Communications Manager (ICM) Unit handles all requests for managing the Incoming Message Log Database Unit. The ICM Unit provides a synchronous message service for clients. The ICM Unit allows for message storage, retrieval, update, notification and processing.

Development Type - Extension of software from Unified Build Version 2.3.0.1

Library - Implemented in the Icm server

4.1.8 Message Processing Routing

Purpose - The Message Processing Routing (MPR) Unit controls the message decoders for incoming ECPN messages. The Mpr Unit handles invocation of the individual message decoders and relays the decode status to the ICM Unit.

Development Type - Extension of existing software from Unified Build Version 2.3.0.1

Library - Implemented in the Mpr application

4.1.9 ISA Decoder

Purpose - The ISA Decoder Unit is responsible for performing envelope validation and GS level breakdown on ANSI X12 ISA messages. Fields from the ISA, IEA, GS, and GE lines are parsed and stored by the ISA Decoder Unit.

Development Type - New development

Library - Implemented in the libX12Dec library archive and the X12IsaDec application

4.1.10 GS Decoder

Purpose - The GS Decoder Unit is responsible for ST level breakdown on GS messages. Fields from the ST and SE lines are parsed by the GS Decoder Unit.

Development Type - New development

Library - Implemented in the libX12Dec library archive and the X12GsDec application

4.1.11 ST Decoder

Purpose - The ST Decoder Unit is responsible for ST level breakdown on GS messages. Fields from various lines are parsed by the ST Decoder Unit.

Development Type - New development

Library - Implemented in the libX12Dec library archive and the X12StDec application

4.1.12 GS Router

Purpose - The GS Router Unit is used to provide GS level message routing for ANSI X12 messages. Multiple GS segments contained within a single ISA segment are divided into separate ISA segments for transmission.

Development Type - New development

Library - Implemented in the libX12Dec library archive, the X12GsDec application, and the ForwardX12 server

4.1.13 Message Forwarder

Purpose - The Message Forwarder Unit is intended to handle X12 destination routing internal to the ECPN. ISA messages are received from the GS Router Unit and destination(s) are determined from the Message Route Database Unit. Various attributes can be used in determining how to route the ISA message.

Development Type -Extension of software from Unified Build Version 2.3.0.1

Library - Implemented in the libComms library archive and the ForwardX12 server

4.1.14 Outgoing Communications Manager

Purpose - The Outgoing Communications Manager (OCM) Unit handles all requests for managing the Outgoing Message Log Database Unit. The OCM Unit provides a synchronous message service for clients. The OCM Unit allows for message storage, retrieval, update, and delivery.

Development Type - Extension of existing software from Unified Build Version 2.3.0.1

Library - Implemented in the Ocm server

4.1.15 Statistics Server

Purpose - The X12 Stats Server Unit is intended to gather statistics about the channels. The statistics are gathered into a report to be used for notification and archival purposes. Accompanying the server is an interface that allows the user to setup notification lists (i.e., email addresses to which notifications will be sent) and to view stats.

Development Type - New development

Library - Implemented in the libX12Stat library archive and the X12StatServe

4.1.16 Remote User Database

Purpose - The Remote User Database Unit is used to define the characteristics of remote users and accounts. Remote users are defined by such parameters as login name, login type (either email or FTP), data directory, and active status.

Development Type - New development

Library - Implemented in the RemoteUserDB application

4.1.17 Interface Database

Purpose - The Interface Database Unit is used to define the characteristics of the ECPN communication interfaces. The communication interfaces are defined by such parameters as interface name, interface type, interface cross reference, node type, message type, cycle rate and hardware device. A specific set of parameters is also defined for each interface type. The interface parameters are configurable through a graphical user interface (GUI). The GUI also displays the current settings/status of the interfaces.

Development Type - Extension of existing software from Unified Build Version 2.3.0.1

Library - Implemented in the EditChannels application

4.1.18 Segment Terminator Database

Purpose - The Segment Terminator Database Unit is intended to provide the user with a means of specifying translation or “pre-processing” operations to apply to messages originating from a specific node. The primary component of the Segment Terminator Database Unit is the database itself. The database consists of a series of ASCII files, one for each database entry, stored in the same directory on a UNIX file system. Each ASCII file contains a description of the operations to be taken for the associated database entry. A graphical user interface is provided so that a user may interact with the database. Through this interface a user can add new segment terminator entries, edit existing entries, delete entries from the database, activate entries, and deactivate entries.

Development Type - New development

Library - Implemented in the SegTermDB application

4.1.19 Statistics Database

Purpose - The Stats Database Unit is used to gather and maintain stats for X12 channel activity. It is also used to store site information used by the X12 Stats Server Unit.

Development Type - New development

Library - Implemented in the libX12Stat library archive

4.1.20 Message Log Database

Purpose - The Message Log Database Unit provides a customizable set of message logs and a GUI for viewing the contents of the logs. There are actually two databases: one for incoming messages and one for outgoing messages. The GUI displays individual attributes for each message in the database and also provides a link to the message's raw data. The Message Log Database Unit is managed through the ICM and OCM Units.

Development Type - Extension of existing software from Unified Build Version 2.3.0.1

Library - Implemented through the LogMgr application and its interface with the ICM and OCM Units

4.1.21 Site Identification Database

Purpose - The Edit Sites DB Unit allows users to add, edit, and delete sites within the Sites database. Each site entry contains an activation flag, a site field, and an ISA field.

Development Type - New development

Library - Implemented in the libComms library archive and the EditSitesDb application

4.1.22 RDBMS Implementation

This section contains information for the following items:

- ◆ RDBMS Database
- ◆ Backside RDBMS Archiver
- ◆ RDBMS Retrieval

4.1.22.1 RDBMS Database

Purpose - The RDBMS Database Unit provides data structures to store the raw message data produced by the Archiver and the Message Log Database produced by the Communications Managers. The structure of the RDBMS ensures that:

- ◆ Each raw message (incoming and outgoing) is directly associated with its decoded ISA, GS, and ST information from the message logs.

- ◆ Each incoming message is directly associated with any outgoing messages that were generated by the GS Router or Message Forwarder.

Development Type - New development

Library - Implemented in the Oracle RDBMS database creation SQL scripts

4.1.22.2 Backside RDBMS Archiver

Purpose - The Backside RDBMS Archiver Unit provides an automated capability to load the Message Log Database into an Oracle Relational Database Management System (RDBMS) for long term storage. This unit executes as a separate process (off-line from the real-time Message Processing) to ensure maximum system throughput. This unit reads the unaltered raw message data produced by the Archiver and the Message Log Database produced by the Incoming Communications Manager and the Outgoing Communications Manager. The raw message and log data is written to the Message Log Oracle Database.

Development Type - New development

Library - Implemented in the Oracle RDBMS loading server

4.1.22.3 RDBMS Retrieval

Purpose - The RDBMS Retrieval Unit provides a GUI interface for operators to enter values for message log fields to direct a retrospective search of the Message Log Oracle Database. This unit incorporates the operator's data values into a dynamic SQL query, executes the query, and presents the retrieved data in a scrolling text window. The operator has the capability to view the raw message file for any selected entry in the scrolling text window or retrieve all outgoing message(s) associated with a given incoming message or the incoming message associated with a given outgoing message.

Development Type - New development

Library - Implemented in the Oracle RDBMS retrieval application

4.1.23 GenWatch

Purpose - GenWatch is a user-oriented network monitoring program that is used to monitor the status of ECPN nodes that are accessible via MILNET/INTERNET. Network connectivity status is obtained via raw ICMP. System-level status of the remote nodes can be obtained via SNMP. The configuration and status of the application software on the ECPN network nodes can also be obtained, allowing the user to view a table of the configurable interfaces, all running processes and the channel status of all running interfaces.

Development Type - Extension of existing software from GenWatch Version 1.2.

Library - Implemented in the GDbm, GWNodes, GWNodeStat, Gtp, GWSysMonMaster, and GWTrace applications

4.1.24 Alert Daemon

Purpose - The Alert Daemon Unit receives system alerts from the ECPN applications. The alerts are stored on disk and delivered to clients upon requests.

Development Type - Extension of existing software from Unified Build Version 2.3.0.1

Library - Implemented in the libAlerts library archive and the AlertDaemon server

4.1.25 Alert Notifier

Purpose - The Alert Notifier Unit receives alerts from the Alert Daemon Unit and performs the notification actions defined in the Alert Notifier Database Unit.

Development Type - New development item

Library - Implemented in the NEPAlertNtfy daemon

4.1.26 Alert Notifier Database

Purpose - The Alert Notifier Database Unit is used to define the notification action(s) to be performed when an alert is received. An entry in the database consists of the alert criteria and one or more notification actions.

Development Type - New development item

Library - Implemented in the AlertNtfyDB application

4.1.27 Message Route Database

Purpose - The Message Route Database Unit contains the ANSI X12 routing entries used by the Message Forwarder Unit. The X12 routing entries define the criteria for determining the destination of an outbound X12 message. A graphical interface is provided for defining the X12 routing entries.

Development Type - Extension of existing software from Unified Build Version 2.3.0.1

Library - Implemented in the X12AFTable application

4.2 Concept of Execution

This section describes the concept of execution among the software units. It includes diagrams and descriptions showing the dynamic relationship of the software units, that is, how they will interact during ECPN operation.

4.2.1 Flow Diagrams

This section depicts in flow diagrams the concept of execution for ECPN. Within these diagrams, each specific step is labeled with a number. Text descriptions of these steps appear following their corresponding numbers in Section 4.2.2 of this document.

4.2.2 Text Descriptions

This section provides a step-by-step description of ECPN data flow and software unit interaction. The step numbers in the following text descriptions correspond directly to the numbered items in the flow diagrams in Section 4.2.1. The flow diagrams depict the steps in their logical order, while the text descriptions provide the details of these steps. All items represented in the flow diagrams appear in Helvetica font in the text descriptions.

1. Passive Communications (Comms) consists of comms channels that receive messages that do not require any explicit action by the NEP. Passive Comms are receive-only.

2. Data that arrives via a comms channel is spooled to an area that is local to that channel (i.e., a directory that is dedicated to receiving communications from a sender).

3. The Remote User Database (DB) contains entries for each sender that can connect passively to the NEP and defines items such as authentication data, as well as identification of the passive sender's personal spool area (see 2).

4. Active Communications (Comms) consists of comms channels that receive and send messages via explicit action by the NEP.

5. Archive In contains the raw data as it was received over a communications channel.

6. The Message Handler performs translation on the raw incoming data.

7. Messages received from VANS and Gateways are usually in X12 ISA format. They may be optionally batched together. For these messages, the Translator is responsible for:

   a. Segment Terminator Translation, resulting in a normalized form that is recognizable by the batch parser (e.g., all lines are terminated with new lines, instead of some sender specific character / character sequence).

   b. Using the Incoming Batch Parser to break down the incoming raw message into individual ISA messages.

   c. Saving each X12 ISA message in In Raw Data (see 13).

   Messages received from AISs are usually in some UDF format. They will not be batched together. For these messages the Translator is responsible for:

   a. Preprocessing the message to remove certain characters that the Sterling Mentor translator cannot handle (e.g., commas are replaced).

   b. Running each individual UDF message through the Sterling Mentor translator to produce an X12 ISA message in accordance with the appropriate standards.

c. Saving each X12 ISA message in In Raw Data (see 13).

8. Each sender has a specific way of indicating line termination and other items. The Segment Terminator Database identifies this correlation (e.g., indicates that each ~ character should be replaced by a line feed for sender AAA).

9. The batch parser breaks down a batch message (i.e., a concatenation of many ISA messages) into individual ISA messages. The batch parser knows how to determine the start/end of ISAs in the batch message by using the Message Bounding DB.

10. The X12 UDF Safestore is a temporary working area for the Sterling Mentor software to read a UDF message.

11. The Sterling Mentor software is a COTS based product for UDF/X12 translation. It also does X12/UDF translation. It works by reading a file and then producing an output file of the requested format.

12. The X12 ISA Safestore is a temporary working area for the Sterling Mentor software to create an X12 ISA message.

13. In Raw Data saves each incoming X12 ISA message and all of its subparts (i.e., GS and ST messages). This is where applications can access the data corresponding to a message entry.

14. After translation, the Incoming Communications Manager is given a new message notification.

15. The Message Processing Router (MPR) is notified of the new message and distributes it to the appropriate decoder process.

16. Envelope validation occurs. This includes validation of the ISA/IEA, GS/GE, and ST/SE message wrappers.

17. The ISA Decoder forwards each X12 ISA message via an interprocess communications channel to the Router Forwarder.

    The ISA Decoder is also responsible for parsing each X12 ISA message into its individual GS messages and saving each of these components in In Raw Data (see 13). The ISA Decoder then passes each individual GS message to the Incoming Communications Manager.

18. A Site Identification (ID) DB check occurs.

19. The GS Decoder parses each X12 GS message into its individual ST messages and saves each of these components in In Raw Data. (See 13) The GS Decoder then passes each individual ST message to the Incoming Communications Manager.

20. The ST Decoder parses the fields (e.g., PO Number) from each ST message. The ST Decoder then passes this information to the Incoming Communications Manager.

21. Individual GS and ST message components are fed to raw data.

22. The Statistics (Stats) DB saves information such as byte counts and originator / recipient information for statistical reporting purposes.

23. The Incoming Communications Manager is responsible for:

    a. Notifying the Message Parser that there is a new message component (ISA or GS) to be parsed.

    b. Updating the Message Logs as information is received. Note that this data can arrive from any of the following five sources:

        - ISA decoder
        - GS decoder
        - ST decoder
        - Translator / validator
        - Message Processor

    c. Handling requests from applications that wish to perform actions in the message logs.

24. The Router Forwarder is responsible for performing GS level routing. Example: One ISA message is received containing two GS messages: GS 1 and GS 2. GS 1 is intended for destination AAA. GS 2 is intended for destination BBB. From the original message the Router Forwarder will produce two ISA messages. ISA 1 will contain GS 1. ISA 2 will contain GS 2.

    Forwarding of each individual ISA message then occurs. Example: Messages intended for PUBLIC are forwarded to all participants, while messages intended for destination BBB are sent only to BBB.

    For each outgoing message, an entry is placed in Out Raw Data (see 26). Note: A message intended for two destinations will contain two copies in Out Raw Data for each message that is going out. An interprocess communications channel is opened with the Outgoing Comms Manager for notification of data ready to transmit.

25. The X12 Routing Table is a one-to-many relationship table that identifies recipients to outgoing circuits. It is used by the Message Forwarder. A routing table match is determined based on criteria such as message sender, message receiver, and input channel.

26. Out Raw Data stores all outgoing messages.

27. The Outgoing Communications Manager is responsible for receiving notifications for data ready to transmit and spooling the data to the required circuit. The Outgoing Communications Manger also handles requests from applications that wish to perform actions in the outgoing message log.

28. The Active Spool Area is local for each comms channel. This directory is polled based on a configurable cycle. The Sterling Mentor software uses the Active Spool Area to read the outgoing X12 ISA messages.

29. The Sterling Mentor software uses the Out UDF Safestore to write the outgoing UDF messages.

30. Each outgoing message is written to Archive Out.

31. The Backside RDBMS Archiver is responsible for reading all message log data and archiving it to the RDBMS DB. Note: Since the RDBMS DB is a background process, normal message processing is not affected by downtime of the RDBMS Archiver during backup / restore / archiving of the RDBMS DB.

32. The RDBMS Retrieval interface provides for queries into the RDBMS database.

33. The Alerts Daemon receives all alerts from the system and stores them in the Alert DB.

34. The Alerts Notifier receives alerts from the Alert Daemon and performs the actions defined in the Alert Notify DB.

4.3 Interface Design

The interface characteristics of the software units will be discussed in the Interface Design Description (IDD) for the ECPN System.

Section 5
CSCI Detailed Design

This section describes each software unit of the ECPN CSCI in the following terms:

- ◆ Unit design decisions
- ◆ Any constraints, limitations, orunusual features in the design of the software unit
- ◆ The programming language to be used

5.1 Passive Comms

Unit Design Decisions - N/A

Constraints/Unusual Features -

- ◆ Configuration of passive communications requires modification of the system password file. Thus, root permission must be granted to the application which performs this configuration.

- ◆ Incoming ftp messages are placed in the local file system based on the login of the ftp user. This directory must correspond to the source directory for the Check Login Unit.

- ◆ Incoming email messages are placed in the local file system by the EmailRecv passive Communications daemon. An email forward file must be properly configured and placed in the login directory of the email user.

Programming Language - Coded in C

5.2 Active Communications

Unit Design Decisions - All active communication interfaces receive their configuration from the Interface Database Unit. Requests for message input and output is controlled by a configurable timer cycle. Transmit and receive status is relayed to the internal Communication Server Units by the communication interface.

Constraints/Unusual Features - The EmailSend communication interface is used for message transmit only. Its counterpart for message receipt is the EmailRecv passive communication interface.

Programming Language - Coded in C

5.3 Check Login

Unit Design Decisions - The Check Login Unit performs checking against the system's list of currently running processes, looking for instances of FTP processes. The list of users is processed, and for each user, if an FTP process is not currently running, the user's directory is processed.

Constraints/Unusual Features - The Check Login Unit is designed to be a single controller for passive communications input. Data that is not actively retrieved by a comms channel will be under the control of this unit.

Programming Language - Coded in C

5.4 Archiver

Unit Design Decisions - Each archived message is stored on disk according to date and key, then is given a random/unique filename. The Archiver writes to this data file in one kilobyte data blocks.

Constraints/Unusual Features - The Archiver Unit uses a configurable environment variable to allow for customization of the archive location.

Programming Language - Coded in C

5.5 X12 Translator

Unit Design Decisions - The X12 Translator Unit performs segment terminator translation by performing byte level deletions and replacements as defined in the Segment Terminator Database Unit. X12 message delimiting is performed by using regular expressions to define start-of-message, end-of-message and end-of-line indicators. Each section of the raw batch data is then injected as either a message or an unrecognized block.

Constraints/Unusual Features -

- The segment terminator translation is limited in its replacement and deletion procedure. Multiple character sets cannot be operated on at once (i.e., n for m replacements or n deletions). The replacements are performed as one-for-one; the deletions are performed on one byte.

- The message delimiting algorithm is performed on recognizable lines. (i.e., carriage returns). Input data that does not delimit lines with carriage returns does not pass through the algorithm correctly.

- The ASCII character set can be used in specifying start-of-message, end-of-message and end-of-line.

Programming Language - Coded in C

5.6 UDF Translator

Unit Design Decisions - The UDF Translator unit performs the translations using part of Sterling Software's Gentran:Mentor software. The translation to be done in the software is dependent upon a Trading Partnership Code. This code is determined from the source and the message type for UDF to X12 translations. The code is determined from the ISA To and ISA From fields of the X12 message for X12 to UDF translations. The list of codes is maintained by the Gentran:Mentor software. The translations themselves are based on maps created and maintained by the Gentran:Mentor software.

Constraints/Unusual Features - The translations are dependent on the correct formation of the Trading Partnership Codes and the translation maps. Translations cannot be performed for sites or message types that do not have Trading Partnership Codes or translation maps existing in the Gentran:Mentor software.

Programming Language - Coded in C

5.7 Incoming Communications Manager

Unit Design Decisions - The ICM Unit is intended to service multiple clients' requests to receive notification of incoming messages and to facilitate the various add, update and delete requests from these clients.

Constraints/Unusual Features - N/A

Programming Language - Coded in C

5.8 Message Processing Routing

Unit Design Decisions - The MPR Unit receives notification of all incoming ECPN messages. Then, based on the type of message (i.e., ISA, GS, ST, etc.), the appropriate message decoder is invoked. Multiple instances of different message decoders can be monitored. The MPR Unit maintains a FIFO queue for each message decoder.

Constraints/Unusual Features -

- The maximum number of entries for the decoder queue is 1000.
- Only one instance of each message decoder can be active at any time.

Programming Language - Coded in C

5.9 ISA Decoder

Unit Design Decisions - The ISA Decoder Unit receives a message to decode from the Message Processing Routing (MPR) Unit and performs envelope validation on the message. Envelope validation is verification that the rules for the ISA, GS and ST wrappers are followed. GS breakdown is then performed. This consists of parsing out the GS lines from the ISA message and submitting them for routing/processing. The ISA Decoder has a configurable linger value that defines how long the decoder will wait for another message from the MPR Unit before it exits.

Constraints/Unusual Features - The ISA Decoder Unit can handle a maximum of 1000 GS messages within a single ISA message, and a maximum size of 64K bytes for each GS message.

Programming Language - Coded in C

5.10 GS Decoder

Unit Design Decisions - The GS Decoder Unit receives a message to decode from the Message Processing Routing Unit and then performs ST level breakdown This consists of parsing out the ST lines from the GS message and submitting them for processing. The GS Decoder has a configurable linger value that defines how long the decoder will wait for another message from the MPR Unit before it exits.

Constraints/Unusual Features - The GS Decoder Unit can handle a maximum of 1000 ST messages within a single GS message, and a maximum size of 64K bytes for each ST message.

Programming Language - Coded in C

5.11 ST Decoder

Unit Design Decisions - The ST Decoder Unit receives a message to decode from the Message Processing Routing Unit and then performs ST level breakdown. The ST Decoder has a configurable linger value that defines how long the decoder will wait for another message from the MPR Unit before it exits.

Constraints/Unusual Features - N/A

Programming Language - Coded in C

5.12 GS Router

Unit Design Decisions - The GS Router Unit separates the GS segments within an ISA segment that contain different GS destinations. Multiple GS segments with the same GS destination will remain in one ISA segment.

Constraints/Unusual Features -

- ◆ GS segments that are separated from the same ISA segment will contain identical ISA lines. This procedure is contrary to the X12 standard but is necessary to fulfill the requirement of GS level routing.

- ◆ The ECPN is not permitted to generate ISA lines at this time. The matching IEA line is adjusted to reflect the GS count for the ISA segment.

Programming Language - Coded in C

5.13 Message Forwarder

Unit Design Decisions - The Message Forwarder Unit can perform destination routing based on the following criteria: input channel, ISA or GS sender, ISA or GS receiver or filename prefix.

Constraints/Unusual Features - The Message Forwarder Unit prevents circular transmission (loops) by not allowing a message to be routed to its input channel.

Programming Language - Coded in C

5.14 Outgoing Communications Manager

Unit Design Decisions - The Outgoing Communications Manager (OCM) Unit is intended to service multiple clients' requests to transmit outgoing messages to destination channels. The OCM Unit facilitates the various add, update and delete requests from these clients.

Constraints/Unusual Features - N/A

Programming Language - Coded in C

5.15 Statistics Server

Unit Design Decisions - The X12 Stats Server Unit maintains statistical data based on the commands issued by its clients. Clients such as the Message Fowarder, Decoder, and Translator Units control the gathering of statistics by issuing the proper commands to the X12 Stat Server Unit. The end result of gathering stats is a report by the server showing information reflecting the execution of the commands issued.

The primary server functions are:

- INCREMENT_*XXX* (*XXX* is either ISA, GS, ST) - keeps a running count of transactions, segments, and messages processed.

- UPDATE STATS - extracts data from the messages that is used for statistical calculations.

- AUTO FORWARD UPDATE - updates the list of channels that shows the destinations that the message was forwarded to.

- CONSOLIDATE - puts data together and add results to ongoing report.

- SEND REPORT - notifies customized list of users with the current report.

- CLEAR CHANNEL - clears statistical data for that channel and restarts statistical calculations.

The X12 StatsServer Unit cycles to clean stats and/or send stats (notifications) according to customizable cycle times. This cycle time is based on increments starting at 00:00:00 (12:00 midnight).

Constraints/Unusual Features -

- The X12 Stats Server Unit is currently referencing the Stats Database Unit but it will be enhanced to reference Oracle as Oracle is incorporated into the system.

- The X12 Stats Interface allows the user to add up to 200 sites and 100 email addresses (for notifications) per site.

Programming Language - Coded in C

5.16 Remote User Database

Unit Design Decisions - Each entry in the Remote User Database Unit is counterchecked against entries in the Interface Database Unit to ensure that no duplicate entries are made and to ensure that each user is assigned a directory name concurrent with an existing data interface.

Constraints/Unusual Features - Entries in the Remote User Database are limited to a maximum of 150.

Programming Language - Coded in C

5.17 Interface Database

Unit Design Decisions - N/A

Constraints/Unusual Features -

- ◆ The Interface Database Unit is designed to prevent duplicate entries and conflicting execution. Duplicate interface names or cross references cannot be entered. More than one interface cannot concurrently use the same hardware device unless the device handles multiplexing (i.e., network interface).

- ◆ The Interface Database Unit is limited to 128 entries.

Programming Language - Coded in C

5.18 Segment Terminator Database

Unit Design Decisions - The Segment Terminator Database Unit actually consists of a series of ASCII data files stored in the same directory on a UNIX file system. This was done for ease of use, but may transition to binary data files some time in the future.

Constraints/Unusual Features -

- ◆ The Segment Terminator Database Unit is limited to 250 entries. Each database entry is further limited to 20 of each operation type, that is 20 deletions and 20 replacements.

- ◆ Currently, there is no provision in the Segment Terminator Database Unit for inserting additional characters.

- ◆ The node name associated with a database entry must be the name of an existing communications channel.

- ◆ Through the graphical user interface, the user must specify the characters to operate on (i.e., Delete or Replace) in octal notation. If the user desires to delete blank lines, then the string “blank_lines” should be entered in the edit text field of the user interface for adding a deletion entry.

Programming Language - Coded in C

5.19 Statistics Database

Unit Design Decisions - One database hierarchy exists for each site that receives/sends data. The Stats Database Unit hierarchy is organized as follows:

Level One - Site
Level Two -
- IN_REPORT - dynamic report for stats on incoming messages
- OUT_REPORT - dynamic report for stats on outgoing messages
- IN_DATA - data extracted from messages that is used in formatting the report
- OUT_DATA - data extracted from messages that is used in formatting the report

Level Three - (data under IN_DATA and OUT_DATA)
- ISA - directory of data collected from each ISA
- ISA _REPORT - report data consolidated from ISA data

Level Four - (data under ISA)
- DESTINATION - directory of data collected from each GS destination
- DESTINATION REPORT - report data consolidated from GS destination data

Level Five - (data under DESTINATION)
- STATS - extracted data from the message
- CHANNEL - list of channels to which the GS was forwarder

Constraints/Unusual Features - N/A

Programming Language - Coded in C

5.20 Message Log Database

Unit Design Decisions - The individual Message Log Database is customizable based on message-type mapping. Message Logs are defined by the message types they hold. The viewing of each message log is customizable through the LogMgr GUI. Various message attributes can be chosen for display while viewing the message log.

Constraints/Unusual Features - The notion of a default message log must always exist in the Message Log Database. The default log is used to handle default mappings for message types which are not mapped into any other message log. The maximum number of message logs is 136. The maximum number of entries in each log is 50,000.

Programming Language - Coded in C

5.21 Site Identification Database

Unit Design Decisions - The Edit Sites DB executable gives the user a graphical interface for manipulating the Sites Database. This database is stored on disk and accessed when decoding an X12 message originating from a Gateway. Each message's ISA sender code is checked against the Sites Database to determine if processing should continue.

Constraints/Unusual Features - The Edit Sites DB can hold 100 site entries.

5Programming Language - Coded in C

5.22 RDBMS Implementation

This section includes information for the following items:

- ◆ RDBMS Database
- ◆ Backside RDBMS Archiver
- ◆ RDBMS Retrieval

5.22.1 RDBMS Database

Unit Design Decisions - The following fields must be defined for all message log records: message log, message type, ISA control number, ISA from, and ISA to.

Constraints/Unusual Features - Initially, the database was created to use the maximum available amount of disk storage provided on the host system. While there is no limit to the number of records that can be stored in the RDBMS, the amount of available disk space limits the
size of the RDBMS.

Programming Language - Coded in SQL

5.22.2 Backside RDBMS Archiver

Unit Design Decisions -

- ◆ The following fields must be defined for all ISA, GS, or ST message log records: message log, message type, ISA control number, ISA from, and ISA to. The following additional fields must be defined for all GS or ST message log records: GS control number, GS from, and GS to. The following additional field must be defined for all ST message log records: ST control number.

- ◆ If an attempt to insert a new message log record into the database encounters an existing record, the existing record is updated.

Constraints/Unusual Features - The environment variables ORACLE_SID and ORACLE_HOME must be defined. If they are not defined, connection to the Oracle RDBMS is not possible.

Programming Language - Coded in C and SQL

5.22.3 RDBMS Retrieval

Unit Design Decisions -

- ◆ The operator may enter data values for the following fields to direct the retrospective search:

  Message log
  Message type

ISA from
ISA to
ISA control number / range
ISA date-time group (DTG) / range
Incoming channel
Outgoing channel
GS from
GS to
GS control number / range
ST type
ST control number / range
Purchase order number
Solicitation control number
Closing DTG / range

- A GUI is provided to ease data entry for the following fields:

Message log
Message type
ISA from
ISA to
Incoming channel
Outgoing channel
GS from
GS to
ST type

Constraints/Unusual Features - The environment variables ORACLE_SID and ORACLE_HOME must be defined. If they are not defined, connection to the Oracle RDBMS is not possible.

Programming Language - Coded in C and SQL

5.23 GenWatch

Unit Design Decisions -

- GenWatch is intended to allow for easy ECPN node status monitoring in an easily interpreted, nontechnical fashion.

- The application software status checking in GenWatch is implemented in a client/server architecture. It allows for easy addition of new application-level checking modules as requirements arise.

Constraints/Unusual Features -

- The GenWatch database is limited to a maximum of 250 entries.

- Duplicate long node names cannot be entered.

- The geographic display and node summary windows may be displayed on multiple work positions on the same computer.

- Making use of a client/server architecture, all GenWatch windows can be displayed on all monitors of all ECPN workstations on a site's Local Area Network.

Programming Language - Coded in C

5.24 Alert Daemon

Unit Design Decisions - Clients to the Alert Daemon Unit request alerts based on their type. The Alert Daemon maintains a status of dismissed or not dismissed for each alert it has received.

Constraints/Unusual Features - The circular queue maintained by the Alert Daemon Unit contains a maximum of 2500 entries.

Programming Language - Coded in C

5.25 Alert Notifier

Unit Design Decisions - The Alert Notifier Unit is capable of performing three types of notification actions:

- Electronic mail which optionally includes the datafile causing the alert

- Personal beeper, pager or cellular phone dialing

- A customizable alert window displayed on the local screen

Constraints / Unusual Features - N / A

Programming Language - Coded in C

5.26 Alert Notifier Database

Unit Design Decisions - The Alert Notifier Database Unit is populated through a user interface that allows for notification methods of electronic mail, beeper dialing and / or local screen alerts.

Constraints / Unusual Features - The Alert Notifier Database is limited to 200 entries. Each entry can contain up to 10 notification actions.

Programming Language - Coded in C

5.27 Message Route Database

Unit Design Decisions -

- ◆ An entry in the Message Route Database Unit is defined by the source channel of the incoming message, a routing field and the destination channel of the outgoing message. The routing field can be sender (FROM), receiver (TO), or filename prefix. Wildcard entries (ALL) also exist for the source channel and routing field. If these are selected, then all messages will pass the message routing criteria for that entry.

- ◆ Individual entries in the Message Route Database Unit can be (de)activated to control their current status during runtime checking.

Constraints / Unusual Features - The Message Route Database Unit is limited to 500 entries.

Programming Language - Coded in C

Section 6
Requirements Traceability

This appendix provides traceability from each software unit identified in this document to the CSCI requirements and, if applicable, software system requirements it addresses.

The software units identified in this plan are as follows:

1. Passive Communications (Comms)
2. Active Comms
3. Check Login
4. Archiver
5. X12 Translator
6. UDF Translator
7. Incoming Comms Manager (ICM)
8. Message Processing Router (MPR)
9. ISA Decoder
10. GS Decoder
11. ST Decoder
12. GS Router
13. Message (MSG) Forwarder
14. Outgoing Comms Manager (OCM)
15. Statistics (Stats) Server
16. Remote User Database (DB)
17. Interface DB
18. Segment Terminator DB
19. Stats DB
20. Message Log DB
21. Site Identification (ID) Database
22. ORACLE
23. GENWatch
24. Alert Daemon
25. Alert Modifier
26. Alert Notify DB
27. Message Route DB

Req No.	Software Unit																										
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
001		X															X										
002	X	X	X													X	X										
003		X															X										
004	X	X	X													X	X										
005		X																									
006																					X						
007			X													X											

008		X											X	X													X
009				X			X							X					X	X							
010	X	X														X											
011		X																									
012		X							X															X	X	X	
013	X																										
014																								X	X	X	
015																								X	X	X	
016																								X	X	X	
017	X	X	X																	X							
018							X							X						X							
019																						X					
020	X	X			X			X	X	X	X	X															
021		X							X	X	X							X	X								
022																							X				
023												X															
024						X																					
025				X			X							X						X							
026																						X					
027									X	X	X																
028		X										X	X														X
029					X													X									

Section 7
Notes

The following acronyms and abbreviations appear in this document:

AIS	Automated Information System
ANSI	American National Standards Institute
ASCII	American Standard Code of Information Interchange
comms	communications
COTS	Commercial Off-the-Shelf
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
DB	Database
DID	Data Item Description
DISA	Defense Information Systems Agency
DoD	Department of Defense
EC/EDI	Electronic Commerce/Electronic Data Interchange
ECPN	Electronic Commerce Processing Node
FIFO	First-In, First-Out
FTP	File Transfer Protocol
GUI	Graphical User Interface
ICM	Incoming Communications Manager
ICMP	Internet Control Message Protocol
ID	Identification
IDD	Interface Design Document
INRI	Inter-National Research Institute
MPR	Message Processing Routing
NEP	Network Entry Point
OCM	Outgoing Communications Manager
PMO	Program Management Office
RDBMS	Relational Database Management System
SDD	Software Design Document
SRS	Software Requirements Specification
UDF	User Defined Format

Appendix A
Tailored DID

Cover - Tailored - INRI has established a standard format for all cover pages for ECPN documents. This format meets the intent of this DID and provides consistency among all ECPN documents.
Title Page - Tailored - INRI has established a standard format for all title pages for ECPN documents. This format meets the intent of this DID and provides consistency among all ECPN documents.
Table of Contents

Section 1 Scope

Section 1.1 Identification

Section 1.2 System Overview

Section 1.3 Document Overview

Section 2.0 Referenced Documents

Section 3.0 CSCI-Wide Design Decisions - Addressed in the Software Requirements Specification (SRS) for ECPN.

Section 4.0 CSCI Architectural Design

Section 4.1 CSCI Components - Tailored - Numbering has been changed to comply with INRI's established format for technical documents. This format meets the intent of this DID and provides consistency among all ECPN documents.

Section 4.2 Concept of Execution

Section 4.3 Interface Design - To be addressed in the Interface Design Description (IDD) for ECPN.

Section 4.3.1 Interface Identification and Diagrams - To be addressed in the Interface Design Description (IDD) for ECPN.

Section 4.3.x (Project-Unique Identifier of Interface) - To be addressed in the Interface Design Description (IDD) for ECPN.

Section 5.0 CSCI Detailed Design

Section 5.x (Project-Unique Identifier of a Software Unit, or Designator of a Group of Software Units)

Section 6.0 Requirements Traceability

Section 7.0 Notes